Präsibert

1.0

Generated by Doxygen 1.8.9.1

Wed Aug 12 2015 09:00:32

Contents

1 Hierarchical Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 Class Documentation **5**

3.1 CameraController Class Reference 5

3.1.1 Detailed Description 6

3.1.2 Constructor & Destructor Documentation 6

3.1.2.1 CameraController 6

3.1.3 Member Function Documentation 6

3.1.3.1 error 6

3.1.3.2 gestureDetected 6

3.1.3.3 onError 6

3.1.3.4 onGestureDetected 6

3.1.3.5 start 7

3.1.3.6 stop 7

3.2 CameraProcessor Class Reference 7

3.2.1 Detailed Description 8

3.2.2 Constructor & Destructor Documentation 8

3.2.2.1 CameraProcessor 8

3.2.2.2 ~CameraProcessor 8

3.2.3 Member Function Documentation 9

3.2.3.1 error 9

3.2.3.2 gestureDetected 9

3.2.3.3 imageReady 9

3.2.3.4 process 9

3.2.3.5 start 9

3.2.3.6 stop 9

3.3 Client Class Reference 10

3.4 Network::ClientSocket Class Reference 11

3.4.1 Detailed Description . . . 12
3.4.2 Constructor & Destructor Documentation . . . 12
3.4.2.1 ClientSocket . . . 12
3.4.2.2 ∼ClientSocket . . . 12
3.4.3 Member Function Documentation . . . 13
3.4.3.1 connectToServer . . . 13
3.4.3.2 disconnectFromServer . . . 13
3.4.3.3 receivedCmd . . . 13
3.4.3.4 receivedData . . . 13
3.4.3.5 sendCmd . . . 13
3.4.3.6 sendData . . . 14
3.5 Network::ConnectedClient Class Reference . . . 15
3.5.1 Detailed Description . . . 16
3.5.2 Constructor & Destructor Documentation . . . 17
3.5.2.1 ConnectedClient . . . 17
3.5.2.2 ∼ConnectedClient . . . 18
3.5.3 Member Function Documentation . . . 18
3.5.3.1 disconnected . . . 18
3.5.3.2 disconnectFromServer . . . 18
3.5.3.3 getClientID . . . 18
3.5.3.4 getPeerAddress . . . 18
3.5.3.5 hasCmdSocket . . . 18
3.5.3.6 hasDataSocket . . . 19
3.5.3.7 newCmd . . . 19
3.5.3.8 newData . . . 19
3.5.3.9 process . . . 19
3.5.3.10 sendCmd . . . 19
3.5.3.11 sendData . . . 19
3.5.3.12 setCmdSocket . . . 20
3.5.3.13 setDataSocket . . . 20
3.6 bb::EM2015::EMaudiorecorder Class Reference . . . 20
3.6.1 Member Function Documentation . . . 21
3.6.1.1 record . . . 21
3.7 ExternalDisplay Class Reference . . . 21
3.8 bb::EM2015::HDMI Class Reference . . . 21
3.9 Message Class Reference . . . 22
3.10 MessageAuthenticator Class Reference . . . 22
3.11 Praesentation Class Reference . . . 23
3.12 Redeanfrage Class Reference . . . 24
3.13 RedeanfrageQueue Class Reference . . . 25

3.14 Network::ServerSocket Class Reference . . . 25
3.14.1 Detailed Description . . . 26
3.14.2 Constructor & Destructor Documentation . . . 27
3.14.2.1 ServerSocket . . . 27
3.14.2.2 ~ServerSocket . . . 27
3.14.3 Member Function Documentation . . . 27
3.14.3.1 beginListening . . . 27
3.14.3.2 clientDisconnect . . . 28
3.14.3.3 closeServer . . . 29
3.14.3.4 disconnectFromClient . . . 29
3.14.3.5 newClient . . . 29
3.14.3.6 newIP . . . 29
3.14.3.7 receivedCmdFromClient . . . 29
3.14.3.8 receivedDataFromClient . . . 29
3.14.3.9 sendCmdToAll . . . 30
3.14.3.10 sendCmdToID . . . 30
3.14.3.11 sendCmdToMultClients . . . 30
3.14.3.12 sendDataToAll . . . 30
3.14.3.13 sendDataToID . . . 30
3.14.3.14 sendDataToMultClients . . . 30
3.15 XMLMessageParser Class Reference . . . 31
3.16 XMLMessageWriter Class Reference . . . 31

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ExternalDisplay 21
bb::EM2015::HDMI 21
QObject
- bb::EM2015::EMaudiorecorder 20
- CameraController 5
- CameraProcessor 7
- Client 10
- Message 22
- MessageAuthenticator 22
- Network::ClientSocket 11
- Network::ConnectedClient 15
- Network::ServerSocket 25
- Praesentation 23
- Redeanfrage 24
- RedeanfrageQueue 25
- XMLMessageParser 31
- XMLMessageWriter 31

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CameraController
    CameraController class 5
CameraProcessor
    CameraProcessor class 7
Client 10
Network::ClientSocket
    ClientSocket class 11
Network::ConnectedClient
    Class for clients connected to the server 15
bb::EM2015::EMaudiorecorder 20
ExternalDisplay 21
bb::EM2015::HDMI 21
Message 22
MessageAuthenticator 22
Praesentation 23
Redeanfrage 24
RedeanfrageQueue 25
Network::ServerSocket
    ServerSocket class 25
XMLMessageParser 31
XMLMessageWriter 31

Chapter 3

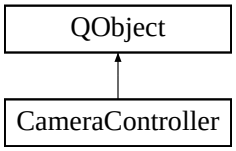
Class Documentation

3.1 CameraController Class Reference

CameraController class.

`#include "Camera/CameraController.hpp"`

Inheritance diagram for CameraController:



Public Slots

- void start ()

  Activates the gesture control feature.
- void stop ()

  Deactivates the gesture control feature.
- void onGestureDetected (int value)

  Gets called when a gesture has been detected.
- void onError (QString e)

  Gets called when an error has occurred.

Signals

- void error (QString e)

  Emitted if an error has occured.
- void gestureDetected (int value)

  Emitted if a gesture is detected.

Public Member Functions

- CameraController (QObject ∗parent=NULL)

  Constructor for the CameraController class.
- virtual ∼CameraController ()

  Deconstructor for the CameraController class.

3.1.1 Detailed Description

CameraController class.

The CameraController class is part of the gesture control component. It works as an interface for the rest of the application. When a CameraController object has been created, it can be used to activate and deactivate the gesture control feature.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 CameraController::CameraController (QObject * *parent* = NULL)

Constructor for the CameraController class.

Parameters

in	*parent*	Parent QObject that creates the CameraController object

This constructor creates a new QThread (but does not start it yet) and checks if the device has a front camera. If not an error signal is emitted.

3.1.3 Member Function Documentation

3.1.3.1 void CameraController::error (QString *e*) [signal]

Emitted if an error has occured.

Parameters

out	*e*	QString with a description of the error

If something is not working (e. g. no front camera available), this signal is emitted. If the CameraController receives an error signal from its CameraProcessor object, this signal is used to transfer that error to the application.

3.1.3.2 void CameraController::gestureDetected (int *value*) [signal]

Emitted if a gesture is detected.

Parameters

out	*value*	Signals which gesture has been detected: -1 = to the left, +1 = to the right

If the CameraProcessor identifies a gesture, it emits a gestureDetected-Signal. This signal here is used to tranfer that signal to the application.

3.1.3.3 void CameraController::onError (QString *e*) [slot]

Gets called when an error has occurred.

Parameters

in	*e*	QString with a description of the error

If an error has occurred, this slot transfers the signal to the application

3.1.3.4 void CameraController::onGestureDetected (int *value*) [slot]

Gets called when a gesture has been detected.

Parameters

in	*value*	Indicates which gesture has been detected: -1 = to the left, +1 = to the right

If a gesture has been detected, this slot transfers the signal to the application

3.1.3.5 void CameraController::start ( ) [slot]

Activates the gesture control feature.

Creates a CameraProcessor object, moves it to a separate thread and calls its start()-function. This starts the scanning process for hand gestures.

3.1.3.6 void CameraController::stop ( ) [slot]

Deactivates the gesture control feature.

Deletes the CameraProcessor object and stops the thread. Thus stops scanning for hand gestures.

The documentation for this class was generated from the following files:

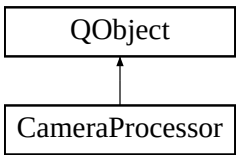
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/CameraController.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Camera/CameraController.cpp

3.2 CameraProcessor Class Reference

CameraProcessor class.

`#include "Camera/CameraProcessor.hpp"`

Inheritance diagram for CameraProcessor:



Public Types

- enum **Status** { **NOTHING**, **RIGHT**, **LEFT** }

Public Slots

- void start ()

  Starts the camera and the gesture detection.
- void stop ()

  Stops the camera and the gesture detection.
- void process ()

  Analyzes the actual frame for a possible gesture.

Signals

- void gestureDetected (int value)

  Emitted if a gesture is detected.
- void error (QString e)

  Emitted if an error has occured.
- void imageReady ()

  Emitted if a new image is ready for processing.

Public Member Functions

- CameraProcessor (QObject *parent=NULL)

  Constructor for the CameraProcessor class.
- virtual ~CameraProcessor ()

  Deconstructor for the CameraController class.

3.2.1 Detailed Description

CameraProcessor class.

The CameraProcessor class is part of the gesture control component. When the gesture control feature gets activated, an instance of this class is created. It then enables the front camera unit and starts the viewfinder. Every frame gets analyzed and if a gesture is detected, an according signal is emitted. When the gesture control feature gets deactivated, this class stops scanning and frees the camera ressource.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 CameraProcessor::CameraProcessor (QObject * *parent* = NULL)

Constructor for the CameraProcessor class.

Parameters

in	*parent*	Parent QObject that creates the CameraProcessor object

This constructor sets all member variables to default values and connects the imageReady-signal with the process()-slot, which is used for analyzing the frames.
To optimize the performance, try changing the following parameters:

- m_framerate: With a framerate of 16 (or less) frames per second every frame gets reliably processed, but maybe it could be set higher for better results.
- m_threshold: The threshold determines when a change in pixel value is deemed significant. The pixel values range from 255 (white) to 0 (black). Too small changes from one frame to the next may be caused by shadows etc., so they should be ignored.
- m_interval_percentage: Determines the percentage of a frame that counts as left or right (where gestures start and finish) and the interval in which a gesture is regarded continued from one frame to the next.

3.2.2.2 CameraProcessor::~CameraProcessor ( ) [virtual]

Deconstructor for the CameraController class.

If the camera ressource is still in use, the stop()-procedure gets called.

3.2.3 Member Function Documentation

3.2.3.1 void CameraProcessor::error (QString *e*) [signal]

Emitted if an error has occured.

Parameters

out	*e*	QString with a description of the error

If an error has occurred (e. g. it was not possible to set a specific resolution), this signal is emitted.

3.2.3.2 void CameraProcessor::gestureDetected (int *value*) [signal]

Emitted if a gesture is detected.

Parameters

out	*value*	Signals which gesture has been detected: -1 = to the left, +1 = to the right

If a gesture has been identified, this signal is emitted.

3.2.3.3 void CameraProcessor::imageReady ( ) [signal]

Emitted if a new image is ready for processing.

When a frame is available, the callback function viewfinder_callback() gets called. If the control variable m_busy is not set, this signal is emitted

3.2.3.4 void CameraProcessor::process ( ) [slot]

Analyzes the actual frame for a possible gesture.

When a frame is available, the callback function viewfinder_callback() gets called. If the control variable m_busy is not set, the signal imageReady() is emitted and process() gets called to analyze the new image which is stored in m_image.

3.2.3.5 void CameraProcessor::start ( ) [slot]

Starts the camera and the gesture detection.

First enables the front camera unit by calling openCamera(), then adjusts the settings and starts the viewfinder by calling startVf().

3.2.3.6 void CameraProcessor::stop ( ) [slot]

Stops the camera and the gesture detection.

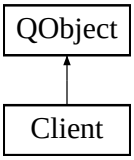
First stops the viewfinder by calling stopVf(), then disables the front camera unit by calling closeCamera().

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/CameraProcessor.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Camera/CameraProcessor.cpp

3.3 Client Class Reference

Inheritance diagram for Client:



Public Types

- enum **LoginState** {
  IDLE, **CONNECTING**, **CONNECTED**, **TRYING**,
  ACCEPTED, **REJECTED** }

Public Slots

- Message * **setSlide** (QMap< QString, QVariant > parameters, QMap< QString, QString > parameter_↩ types)
- Message * **parsePraesentation** (QMap< QString, QVariant > parameters, QMap< QString, QString > parameter_types)
- Message * **loginResponse** (QMap< QString, QVariant > parameters, QMap< QString, QString > parameter_types)
- Message * **stopPraesentation** (QMap< QString, QVariant > parameters, QMap< QString, QString > parameter_types)
- Q_INVOKABLE bb::cascades::Image **getSlide** ()
- Q_INVOKABLE QString **getLastSentMsg** ()
- Q_INVOKABLE QString **getLoginState** ()
- Q_INVOKABLE QString **getBasepath** ()
- Q_INVOKABLE void **requestSlideChange** (int offset)
- Q_INVOKABLE void **requestSlideChangeAbsolute** (int slide)
- Q_INVOKABLE void **sendArbitraryCommand** (QString cmd)
- void **connectionLost** ()
- Q_INVOKABLE void **deliverRecording** (QString path)
- Q_INVOKABLE void **invokeRemote** (Message *msg)
- Q_INVOKABLE void **invokeRemote** (Message *msg, bool cleanup)
- Q_INVOKABLE void **login** ()
- Q_INVOKABLE void **logout** ()
- Q_INVOKABLE void **connectToServer** (QString addr, QString cmd_port, QString data_port)
- void **onNewSlideUrl** (QUrl url)

Signals

- void **slideChanged** (bb::cascades::Image img)
- void **slideChangedUrl** (QUrl url)
- void **messageSent** ()
- void **loginStateChanged** ()
- void **wait** (bool active)
- void **praesentationReady** ()
- void **noMoreSlides** ()

Protected Attributes

- QMap< QString, remoteFunction > **registerdFunctions**
- bb::cascades::Image **m_slide**
- XMLMessageParser * **xmlmp**
- XMLMessageWriter * **xmlmw**
- XMLMessageParser * **xmlmp_data**
- XMLMessageWriter * **xmlmw_data**
- QString **lastSentMsg**
- Network::ClientSocket * **cs**
- LoginState **login_state**
- Praesentation * **prs**
- QString **id**
- bb::EM2015::HDMI * **hdmi**

The documentation for this class was generated from the following files:

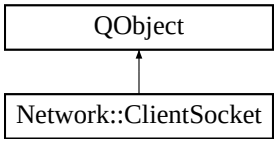
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/Client.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Client/Client.cpp

3.4 Network::ClientSocket Class Reference

ClientSocket class.

`#include "Network/ClientSocket.h"`

Inheritance diagram for Network::ClientSocket:



Public Slots

- bool connectToServer (QString ipAddr_str, QString cmdPort_str, QString dataPort_str)

  Method for connecting the sockets to a server at the specified IP-Address.
- void disconnectFromServer ()

  Method to disconnect from server.
- int sendCmd (QByteArray data)

  Method that is used to send a command from the command socket.
- int sendData (QByteArray data)

  Method that is used to send data from the data socket.

Signals

- void connectedToCmdServer ()

  Signal that is emitted, when when the command socket successfully connected to its server.
- void connectedToDataServer ()

  Signal that is emitted, when when the data socket successfully connected to its server.
- void receivedCmd (QByteArray data)

Signal that is emitted, when a new command is available at the command socket.

- void receivedData (QByteArray data)

  Signal that is emitted, when new data is available at the data socket.

- void lostConnection ()

  Signal that is emitted, when the connection was lost to one the server sockets.

Public Member Functions

- ClientSocket (QObject ∗)

  Constructor of the ClientSocket class.

- virtual ∼ClientSocket ()

  Destructor of the ClientSocket class.

3.4.1 Detailed Description

ClientSocket class.

Instantiates two TCP Client Sockets (command and data) that can connect to servers.
It uses 32 bit integers for determining the length of sent and received data.
The class provides several signals and slots for connection and data handling:

- signals:
  - connectedToCmdServer(): Emitted, when the command socket successfully connected to the server.
  - connectedToDataServer(): Emitted, when the data socket successfully connected to the server.
  - receivedCmd(): Emitted with ByteArray, when a new command is available.
  - receivedData(): Emitted with ByteArray, when new data is available.
  - lostConnection(): Emitted, when the connection to one of the servers (cmd or data) is lost.
- slots:
  - connectToServer(): Tries to establish a connection to a server with the IP and two ports (command and data port) that are given as parameter in QString format.
  - sendCmd(): Sends the command that is given as parameter to the command socket of the server.
  - sendData(): Sends the data that is given as parameter to the data socket of the server.
  - disconnectFromServer(): Disconnects the current connection to the server for both sockets.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Network::ClientSocket::ClientSocket (QObject ∗ *parent*)

Constructor of the ClientSocket class.

Initializes the command and data socket and connects signals and slots for connection and data handling.
Connects the signal *connected()* of the sockets with the handlers (*connectedToCmdServer()* and (*connectedTo↩ DataServer()*) of this class.
Connects the signal *disconnected()* of the sockets with the slot *disconnectFromServer()* of this class.
Connects the signal *readyRead()* of the command socket with the slot *handleNewCmd()* of this class. Connects the signal *readyRead()* of the data socket with the slot *handleNewData()* of this class.

3.4.2.2 Network::ClientSocket::∼ClientSocket ( ) [virtual]

Destructor of the ClientSocket class.

Closes the sockets and deletes them.

3.4.3 Member Function Documentation

3.4.3.1 bool Network::ClientSocket::connectToServer (QString *ipAddr_str,* QString *cmdPort_str,* QString *dataPort_str*) [slot]

Method for connecting the sockets to a server at the specified IP-Address.

Parameters

in	*ipAddr_str*	IP-Address in QString format to connect to.
in	*cmdPort_str*	Specified port for the command connection in QString format.
in	*dataPort_str*	Specified port for the data connection in QString format.

Returns

Returns true, if the connection was established successfully. Otherwise returns false.

This method tries to connect to the server sockets at the IP-Address and the ports that were given as parameters. The method calls the disconnectFromServer() method, if the connection cannot be established.

3.4.3.2 void Network::ClientSocket::disconnectFromServer ( ) [slot]

Method to disconnect from server.

This method lets the socket disconnect from the server that it is connected to.
Emits the *lostConnection*-Signal after it disconnected from the server.

3.4.3.3 void Network::ClientSocket::receivedCmd (QByteArray *data*) [signal]

Signal that is emitted, when a new command is available at the command socket.

Parameters

out	*data*	Command that was read from the socket in QByteArray format.

3.4.3.4 void Network::ClientSocket::receivedData (QByteArray *data*) [signal]

Signal that is emitted, when new data is available at the data socket.

Parameters

out	*data*	Data that was read from the socket in QByteArray format.

3.4.3.5 int Network::ClientSocket::sendCmd (QByteArray *data*) [slot]

Method that is used to send a command from the command socket.

Parameters

in	*data*	Command that is send to the server.

Returns

Returns the number of bytes that were actually send to the server.

This method sends data from the command socket to the server.
A 32 bit integer with information about the data length is send first. Then the actual data follows.

3.4.3.6 int Network::ClientSocket::sendData (QByteArray *data*) [slot]

Method that is used to send data from the data socket.

Parameters

in	*data*	Data that is send to the server.

Returns

Returns the number of bytes that were actually send to the server.

This method sends data from the data socket to the server.
A 32 bit integer with information about the data length is send first. Then the actual data follows.

The documentation for this class was generated from the following files:

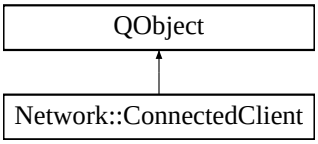
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/ClientSocket.h
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Network/ClientSocket.cpp

3.5 Network::ConnectedClient Class Reference

Class for clients connected to the server.

`#include "Network/ConnectedClient.h"`

Inheritance diagram for Network::ConnectedClient:



Public Slots

- void process ()

  Method is called on start of the Thread.

Signals

- void newCmd (QByteArray data, uint clientID)

  Signal that is emitted, whe an newcommanda is available fromthe commanda socket of the client.
- void newData (QByteArray data, uint clientID)

  Signal that is emitted, when new data is available from the data socket of the client.
- void disconnected (uint clientID)

  Signal that is emitted, when the client was disconnected.
- void finished ()

  Signal that is emitted, when the client is finished and ready for deletion.

Public Member Functions

- ConnectedClient (uint clientID)

  Constructor of the ConnectedClient class.
- virtual ~ConnectedClient ()

  Destructor of the ConnectedClient class.
- void setCmdSocket (QTcpSocket *tcpSocket)

Method used to set the command socket.

- void setDataSocket (QTcpSocket *tcpSocket)

  Method used to set the data socket.

- bool hasCmdSocket ()

  Method that returns, whether the command socket is established already.

- bool hasDataSocket ()

  Method that returns, whether the data socket is established already.

- int sendCmd (QByteArray data)

  Sends a command to the connected client.

- int sendData (QByteArray data)

  Sends data to the connected client.

- void disconnectFromServer ()

  Closes the connection to the Server.

- uint getClientID ()

  Returns the clientID of the socket.

- QHostAddress getPeerAddress ()

  Returns the peer address of the connected client.

3.5.1 Detailed Description

Class for clients connected to the server.

Class for clients that connect to the Server Socket (ServerSocket class).
All of the objects that are created from this class are stored in an individual thread that is started when a new connection is established.
It uses 32 bit integers for determining the length of sent and received data.
The class provides several functions, signals and slots that are used to exchange data with a client:

- functions:
  - setCmdSocket(): Sets the socket that is given as parameter to the command socket.
  - setDataSocket(): Sets the socket that is given as parameter to the data socket.
  - hasCmdSocket(): Returns true, if the command socket is set up.
  - hasDataSocket(): Returns true, if the data socket is set up.
  - sendCmd(): Sends the ByteArray that is given as parameter to the clients command socket.
  - sendData(): Sends the ByteArray that is given as parameter to the clients data socket.
  - disconnectFromServer(): Disconnects both sockets from the servers sockets.
  - getClientID(): Returns the ID of the client.
  -
  - getPeerAddress(): Returns the peer address of the client.
- signals:
  - newCmd(): Emitted with the clientID and data when a new command is available from the clients command socket.
  - newData(): Emitted with the clientID and data when new data is available from the clients data socket.
  - disconnected(): Emitted with the clientID, when the connection to one of the clients sockets is lost.
  - finished(): Emitted, when the connection to the client is lost and the client socket can be destroyed.
- slots:
  - process(): This slot is connected to the start signal of the thread that stores the ConnectedClient object.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Network::ConnectedClient::ConnectedClient (uint *clientID*)

Constructor of the ConnectedClient class.

Parameters

in	*clientID*	ID of the client that is created.

Initializes the variables *m_clientID* with the value that was given as parameter.
Also initializes the booleans *hasCmdSocket* and *hasDataSocket* with the value *false* and *m_next_block_size_cmd* and *m_next_block_size_data* with the value *0*.
The constructor does not implement a parent object so that it can be moved into a QThread.

3.5.2.2 Network::ConnectedClient::~ConnectedClient ( ) [virtual]

Destructor of the ConnectedClient class.

The destructor closes the sockets and deletes them.

3.5.3 Member Function Documentation

3.5.3.1 void Network::ConnectedClient::disconnected (uint *clientID*) [signal]

Signal that is emitted, when the client was disconnected.

Parameters

out	*clientID*	ID of the client that lost the connection.

3.5.3.2 void Network::ConnectedClient::disconnectFromServer ( )

Closes the connection to the Server.

This class closes the connection to the servers sockets.

3.5.3.3 uint Network::ConnectedClient::getClientID ( )

Returns the clientID of the socket.

Returns

Returns the clientID of the socket.

3.5.3.4 QHostAddress Network::ConnectedClient::getPeerAddress ( )

Returns the peer address of the connected client.

Returns

Returns the peer address of the connected client in QHostAddress format.

This function returns the peer address of the connected client.
The command socket is primarily used to determine the peer address.
If the command socket is not available, the data socket is used.
If none of the sockets is available, localhost is returned as peer address.

3.5.3.5 bool Network::ConnectedClient::hasCmdSocket ( )

Method that returns, whether the command socket is established already.

Returns

Returns true, if the socket is set up and available.

3.5.3.6 bool Network::ConnectedClient::hasDataSocket ( )

Method that returns, whether the data socket is established already.

Returns

Returns true, if the socket is set up and available.

3.5.3.7 void Network::ConnectedClient::newCmd (QByteArray *data,* uint *clientID*) [signal]

Signal that is emitted, whe an newcommanda is available fromthe commanda socket of the client.

Parameters

out	*data*	Data in QByteArray format that is send out.
out	*clientID*	Own ID of the client that sends the data.

3.5.3.8 void Network::ConnectedClient::newData (QByteArray *data,* uint *clientID*) [signal]

Signal that is emitted, when new data is available from the data socket of the client.

Parameters

out	*data*	Data in QByteArray format that is send out.
out	*clientID*	Own ID of the client that sends the data.

3.5.3.9 void Network::ConnectedClient::process ( ) [slot]

Method is called on start of the Thread.

This method is called on start of the QThread that the ConnectedClient object was moved to.

3.5.3.10 int Network::ConnectedClient::sendCmd (QByteArray *data*)

Sends a command to the connected client.

Parameters

in	*data*	Command in QByteArray format that is send.

Returns

Returns the amount of bytes that were actually sent to the client.

This method is used to send a command to the connected client that is given in QByteArray format as parameter. A 32 bit integer with information about the data length is send first. Then the actual data follows.

3.5.3.11 int Network::ConnectedClient::sendData (QByteArray *data*)

Sends data to the connected client.

Parameters

in	*data*	Data in QByteArray format that is send.

Returns

Returns the amount of bytes that were actually sent to the client.

This method is used to send data to the connected client that is given in QByteArray format as parameter.
A 32 bit integer with information about the data length is send first. Then the actual data follows.

3.5.3.12 void Network::ConnectedClient::setCmdSocket (QTcpSocket * *tcpSocket*)

Method used to set the command socket.

Parameters

in	*tcpSocket*	New socket object that the clients command socket is assigned to.

Each time a new connection is established with the server, a new QTcpSocket object is created.
This object is then passed to this function as parameter.
Within this function, the new socket object is then assigned to the command socket.

3.5.3.13 void Network::ConnectedClient::setDataSocket (QTcpSocket * *tcpSocket*)

Method used to set the data socket.

Parameters

in	*tcpSocket*	New socket object that the clients data socket is assigned to.

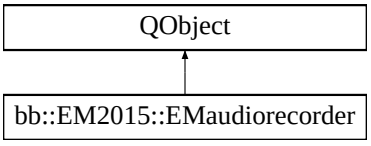
Each time a new connection is established with the server, a new QTcpSocket object is created.
This object is then passed to this function as parameter.
Within this function, the new socket object is then assigned to the data socket.

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/ConnectedClient.h
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Network/Connected↩ Client.cpp

3.6 bb::EM2015::EMaudiorecorder Class Reference

Inheritance diagram for bb::EM2015::EMaudiorecorder:



Public Member Functions

- char * record ()

  Initalizes the EMaudiorecorder.
- unsigned int **stop** ()
- void **LED_TEST** ()

Public Attributes

- bool **armed**
- bool **record_running**
- int **current_file**

3.6.1 Member Function Documentation

3.6.1.1 char * bb::EM2015::EMaudiorecorder::record ( )

Initalizes the EMaudiorecorder.

Initalizes the EMaudiorecorder: global variables and classes.

Parameters

-none-	

Returns

-none-

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/EMaudiorecorder.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/AudioRecorder/EMaudiorecorder.cpp

3.7 ExternalDisplay Class Reference

Public Member Functions

- int **open** ()
- int **close** ()
- void **setResolution** (RESOLUTIONS_T res)
- RESOLUTIONS_T **getResolution** ()
- int **showImage** (bb::ImageData imageData)

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/ExternalDisplay.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/HDMI/ExternalDisplay.cpp

3.8 bb::EM2015::HDMI Class Reference

Public Member Functions

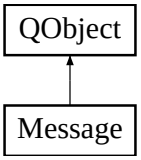
- **HDMI** (RESOLUTIONS_T hdmi_resolution)
- void **show_slide** (QUrl img_url)
- void **show_last_slide** ()

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/HDMI.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/HDMI/HDMI.cpp

3.9 Message Class Reference

Inheritance diagram for Message:



Public Member Functions

- **Message** (QString command, QString sender, QString receiver)
- QString **getCommand** ()
- QString **getSender** ()
- QString **getReceiver** ()
- const QMap< QString, QVariant > * **getParameters** ()
- const QMap< QString, QString > * **getParameterTypes** ()
- void **setParameterList** (QMap< QString, QVariant > list)
- void **setParameterTypeList** (QMap< QString, QString > types)
- int **addParameter** (QString name, QString value)
- int **addParameter** (QString name, QDateTime value)
- int **addParameter** (QString name, int value)
- int **addParameter** (QString name, double value)
- int **addParameter** (QString name, QByteArray value)
- QDateTime **getTimestamp** ()
- void **setTimestamp** (QDateTime ts)

Friends

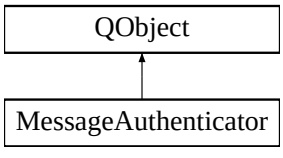
- class **XMLMessageParser**
- class **XMLMessageWriter**
- class **Client**
- class **Praesentation**

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/Message.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Message/Message.cpp

3.10 MessageAuthenticator Class Reference

Inheritance diagram for MessageAuthenticator:



Public Slots

- void **authenticateMessage** (QByteArray msg)
- void **setKey** (QByteArray key)

Signals

- void **messageAuthenticated** (QByteArray msg)

Public Member Functions

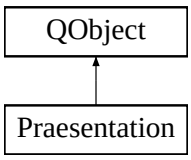
- QByteArray **hmacSha1** (QByteArray baseString)
- QByteArray **hmacSha1** (QByteArray key, QByteArray baseString)

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/MessageAuthenticator.↩
h
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Message/Authentication/Message↩
Authenticator.cpp

3.11 Praesentation Class Reference

Inheritance diagram for Praesentation:



Public Slots

- void **parsePraesentation** (QMap< QString, QVariant > params, QMap< QString, QString > types)
- Message * **packPraesentation** ()
- Message * **packPraesentation** (Message *msg)
- void **appendSlide** (QString path)
- int **getCurrentSlide** ()
- int **getTotalSlides** ()
- void **setSlide** (int slide)
- QString **getPraesentationId** ()
- QString **getBasepath** ()
- void **reset** ()
- void **stop** ()

Signals

- void **slideChanged** (bb::cascades::Image)
- void **slideChangedUrl** (QUrl url)
- void **praesentationParsed** (Message *response)
- void **parsing** (bool active)

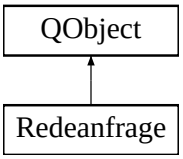
- void **praesentationReady** ()
- void **isRunning** (bool active)

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/Praesentation.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Praesentation/Praesentation.↩ cpp

3.12 Redeanfrage Class Reference

Inheritance diagram for Redeanfrage:



Public Types

- enum **RedeanfrageState** {
  PREPARATION, **QUEUED**, **ACCEPTED**, **REJECTED**,
  FINISHED }

Public Slots

- void **prepare** ()
- void **queue** (QString clientId)
- void **queue** ()
- void **accept** ()
- void **reject** ()
- void **finish** ()
- Message ∗ **packRedeanfrage** ()
- QString **getClientId** ()
- void **setClientId** (QString clientId)

Signals

- void **stateChanged** (QString state)

Public Member Functions

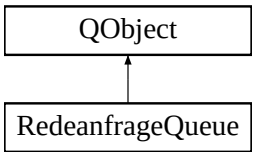
- **Redeanfrage** (QString clientId)

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/Redeanfrage.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Redeanfrage/Redeanfrage.↩ cpp

3.13 RedeanfrageQueue Class Reference

Inheritance diagram for RedeanfrageQueue:



Public Slots

- int **enqueue** (Redeanfrage *ranf)
- Redeanfrage * **dequeue** ()
- void **clear** ()
- int **getSize** ()
- QString **getClientIdAt** (int i)

Signals

- void **sizeChanged** (int size)

The documentation for this class was generated from the following files:

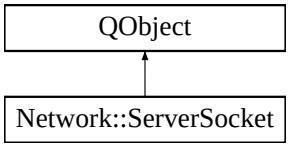
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/Redeanfrage↩ Queue.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Redeanfrage/Redeanfrage↩ Queue.cpp

3.14 Network::ServerSocket Class Reference

ServerSocket class.

`#include "Network/ServerSocket.h"`

Inheritance diagram for Network::ServerSocket:



Public Slots

- bool beginListening (QString cmdPort_str, QString dataPort_str)

  Method that is called to start listening for incoming connections.
- void closeServer ()

  Signal to close the server.
- bool disconnectFromClient (uint clientID)

  Method to close the connection to a client.
- void sendCmdToAll (QByteArray data)

Method for sending a command to all clients.

- void sendDataToAll (QByteArray data)

  Method for sending data to all clients.
- void sendCmdToMultClients (QByteArray data, QList< uint > clientIDs)

  Method for sending a command to a specified list of clients.
- void sendDataToMultClients (QByteArray data, QList< uint > clientIDs)

  Method for sending data to a specified list of clients.
- int sendCmdToID (QByteArray data, uint clientID)

  Method for sending a command to a client with specified ID.
- int sendDataToID (QByteArray data, uint clientID)

  Method for sending data to a client with specified ID.

Signals

- void newIP (QString newIP)

  Signal that is emitted, when the server is set up correctly and a correct IP was found.
- void clientDisconnect (uint clientID)

  Signal that is emitted, when the connection to a client was lost.
- void stoppedServer ()

  Signal that is emitted, when the server was stopped.
- void receivedCmdFromClient (QByteArray data, uint clientID)

  Signal that is emitted, when a new command was received from a client.
- void receivedDataFromClient (QByteArray data, uint clientID)

  Signal that is emitted, when new data was received from a client.
- void newClient (uint clientID)

  Signal that is emitted, when a new client connected to the server.

Public Member Functions

- ServerSocket (QObject *parent)

  Constructor of the ServerSocket class.
- virtual ~ServerSocket ()

  Destructor of the ServerSocket class.

3.14.1 Detailed Description

ServerSocket class.

Instantiates two TCP Server Sockets (command and data) that clients can connect to for communication.
The class provides several signals and slots for connection and data handling:

- signals:
  - newIP(): Emitted with current IP, when the server is set up.
  - clientDisconnect(): Emitted with clientID, when connection to a client is lost.
  - stoppedServer(): Emitted, when server is stopped.
  - receivedCmdFromClient(): Emitted with data and clientID, when a command was received from a client.
  - receivedDataFromClient(): Emitted with data and clientID, when data was received from a client.
- slots:

- beginListening(): Calling this slot makes the server start to listen for incoming connections (command and data) of the ports that are given as parameter.
- closeServer(): Shuts down the server.
- sendCmdToAll(): Sends the command that is given as parameter to all of its clients command sockets.
- sendDataToAll(): Sends the data that is given as parameter to all of its clients data sockets.
- sendCmdToID(): Sends the command that is given as parameter to the client with the specified ID.
- sendDataToID(): Sends the data that is given as parameter to the client with the specified ID.
- sendCmdToMultClients(): Sends the command that is given as parameter to the IDs of clients that are given in a QList as parameter.
- sendCmdToMultClients(): Sends the data that is given as parameter to the IDs of clients that are given in a QList as parameter.
- disconnectFromClient(): Closes the connection to the client whose ID is given as parameter.

The ServerSocket manages all of the connected clients in a list (m_clientList) with a specific ID. For each client that establishes a connection to the server, an object of the *ConnectedClient*-class is created, pushed to an own thread and stored in m_clientList.
The *ConnectedClient*-class contains two QTcpSockets that the server can use to communicate with it's clients.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 Network::ServerSocket::ServerSocket (QObject * *parent*)

Constructor of the ServerSocket class.

Parameters

in	*parent*	Parent QObject that creates the server socket object.

The constructor initializes *m_clientID* (ID that is given to client) with its initial value *0*.
Also initializes both types of server sockets with *parent* as parameter.

3.14.2.2 Network::ServerSocket::~ServerSocket ( ) [virtual]

Destructor of the ServerSocket class.

Closes the server socket and deletes itself.

3.14.3 Member Function Documentation

3.14.3.1 bool Network::ServerSocket::beginListening (QString *cmdPort_str*, QString *dataPort_str*) [slot]

Method that is called to start listening for incoming connections.

Parameters

in	*cmdPort_str*	Listening port for incoming command connections in QString format.
in	*dataPort_str*	Listening port for incoming data connections in QString format.

Returns

Returns true, if the listening for incoming connections started successfully.

First the IP-Address of the server is located and the signal *newIP()* with the IP in QString format is emitted.
If no IP was found, localhost is used as IP-Address.
Afterwards initializes the TCP server sockets and starts listening for incoming connections on any address.
Also connects the signals *newConnection* of the server sockets with the handler slot *handleNewConnection()*.

3.14.3.2 void Network::ServerSocket::clientDisconnect (uint *clientID*) [signal]

Signal that is emitted, when the connection to a client was lost.

Parameters

out	*clientID*	ID of the client that the connection was lost to.

3.14.3.3 void Network::ServerSocket::closeServer ( ) [slot]

Signal to close the server.

Disconnects from all of the clients in *m_clientList* and closes both servers. Emits the signal *stoppedServer* afterwards.

3.14.3.4 bool Network::ServerSocket::disconnectFromClient (uint *clientID*) [slot]

Method to close the connection to a client.

Parameters

in	*clientID*	This method terminates the connection to a client that is connected to the server.

3.14.3.5 void Network::ServerSocket::newClient (uint *clientID*) [signal]

Signal that is emitted, when a new client connected to the server.

Parameters

out	*clientID*	ID of the client that connected to the server.

This Signal is only emitted, when **both** types of sockets (data and command) successfully connected to the server.

3.14.3.6 void Network::ServerSocket::newIP (QString *newIP*) [signal]

Signal that is emitted, when the server is set up correctly and a correct IP was found.

Parameters

out	*newIP*	IP-Address in QString format that was found.

3.14.3.7 void Network::ServerSocket::receivedCmdFromClient (QByteArray *data,* uint *clientID*) [signal]

Signal that is emitted, when a new command was received from a client.

Parameters

out	*data*	Command that was received from the client.
out	*clientID*	ID of the client that sent the command.

3.14.3.8 void Network::ServerSocket::receivedDataFromClient (QByteArray *data,* uint *clientID*) [signal]

Signal that is emitted, when new data was received from a client.

Parameters

out	*data*	Data that was received from the client.
out	*clientID*	ID of the client that sent the data.

3.14.3.9 void Network::ServerSocket::sendCmdToAll (QByteArray *data*) [slot]

Method for sending a command to all clients.

Parameters

in	*data*	Command that is send to the clients.

Sends a command to all clients.

3.14.3.10 int Network::ServerSocket::sendCmdToID (QByteArray *data,* uint *clientID*) [slot]

Method for sending a command to a client with specified ID.

Parameters

in	*data*	Command that is send to the clients.
in	*clientID*	ID of the client that the command is send to.

Sends a command to the client with the specified ID.

3.14.3.11 void Network::ServerSocket::sendCmdToMultClients (QByteArray *data,* QList< uint > *clientIDs*) [slot]

Method for sending a command to a specified list of clients.

Parameters

in	*data*	Command that is send to the clients.
in	*clientIDs*	QList with clientIDs to which the data is send.

Sends a command to all clients whose clientIDs are specified in the QList.

3.14.3.12 void Network::ServerSocket::sendDataToAll (QByteArray *data*) [slot]

Method for sending data to all clients.

Parameters

in	*data*	Data that is send to the clients.

Sends data to all clients.

3.14.3.13 int Network::ServerSocket::sendDataToID (QByteArray *data,* uint *clientID*) [slot]

Method for sending data to a client with specified ID.

Parameters

in	*data*	Data that is send to the clients.
in	*clientID*	ID of the client that the data is send to.

Sends data to the client with the specified ID.

3.14.3.14 void Network::ServerSocket::sendDataToMultClients (QByteArray *data,* QList< uint > *clientIDs*) [slot]

Method for sending data to a specified list of clients.

Parameters

in	*data*	Data that is send to the clients.
in	*clientIDs*	QList with clientIDs to which the data is send.

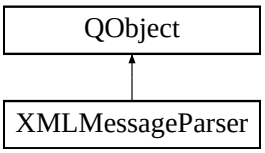
Sends data to all clients whose clientIDs are specified in the QList.

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/ServerSocket.h
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Network/ServerSocket.cpp

3.15 XMLMessageParser Class Reference

Inheritance diagram for XMLMessageParser:



Public Slots

- void **parseMessage** (QByteArray bytes)
- void **parseCmdMessage** (QByteArray bytes)
- void **parseDataMessage** (QByteArray bytes)
- void **parseCmdMessage** (QByteArray bytes, uint clientId)
- void **parseDataMessage** (QByteArray bytes, uint clientId)

Signals

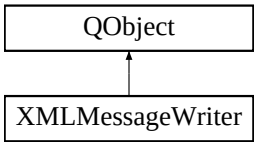
- void **messageParsed** (Message ∗msg)
- void **cmdMessageParsed** (Message ∗msg)
- void **dataMessageParsed** (Message ∗msg)
- void **cmdMessageParsed** (Message ∗msg, uint clientId)
- void **dataMessageParsed** (Message ∗msg, uint clientId)

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/XMLMessageParser.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Message/XML/XMLMessageParser.cpp

3.16 XMLMessageWriter Class Reference

Inheritance diagram for XMLMessageWriter:



Public Slots

- void **writeMessage** (Message ∗msg)
- void **writeCmdMessage** (Message ∗msg)
- void **writeDataMessage** (Message ∗msg)
- void **writeCmdMessage** (Message ∗msg, uint clientId)
- void **writeDataMessage** (Message ∗msg, uint clientId)
- void **writeCmdMessage** (Message ∗msg, QList< uint > clientIDs)
- void **writeDataMessage** (Message ∗msg, QList< uint > clientIDs)

Signals

- void **messageWritten** (QByteArray msg)
- void **cmdMessageWritten** (QByteArray msg)
- void **dataMessageWritten** (QByteArray msg)
- void **cmdMessageWritten** (QByteArray msg, uint clientId)
- void **dataMessageWritten** (QByteArray msg, uint clientId)
- void **cmdMessageWritten** (QByteArray msg, QList< uint > clientIDs)
- void **dataMessageWritten** (QByteArray msg, QList< uint > clientIDs)

The documentation for this class was generated from the following files:

- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/include/XMLMessage↩ Writer.hpp
- C:/Users/Niklas/Desktop/EM2015_PraesiBert/Common/ClientServerShareLib/src/Message/XML/XML↩ MessageWriter.cpp